



Server-Side Eclipse



Martin Lippert
akquinet agile GmbH
martin.lippert@akquinet.de



Outline

- Introduction
- Why Eclipse?
- Different Opportunities
 - Pure OSGi
 - OSGi and Plug-In Runtime
 - Pure Plug-In Runtime
 - Eclipse Headless
 - OSGi in a Web-Container
 - Web-Server inside OSGi



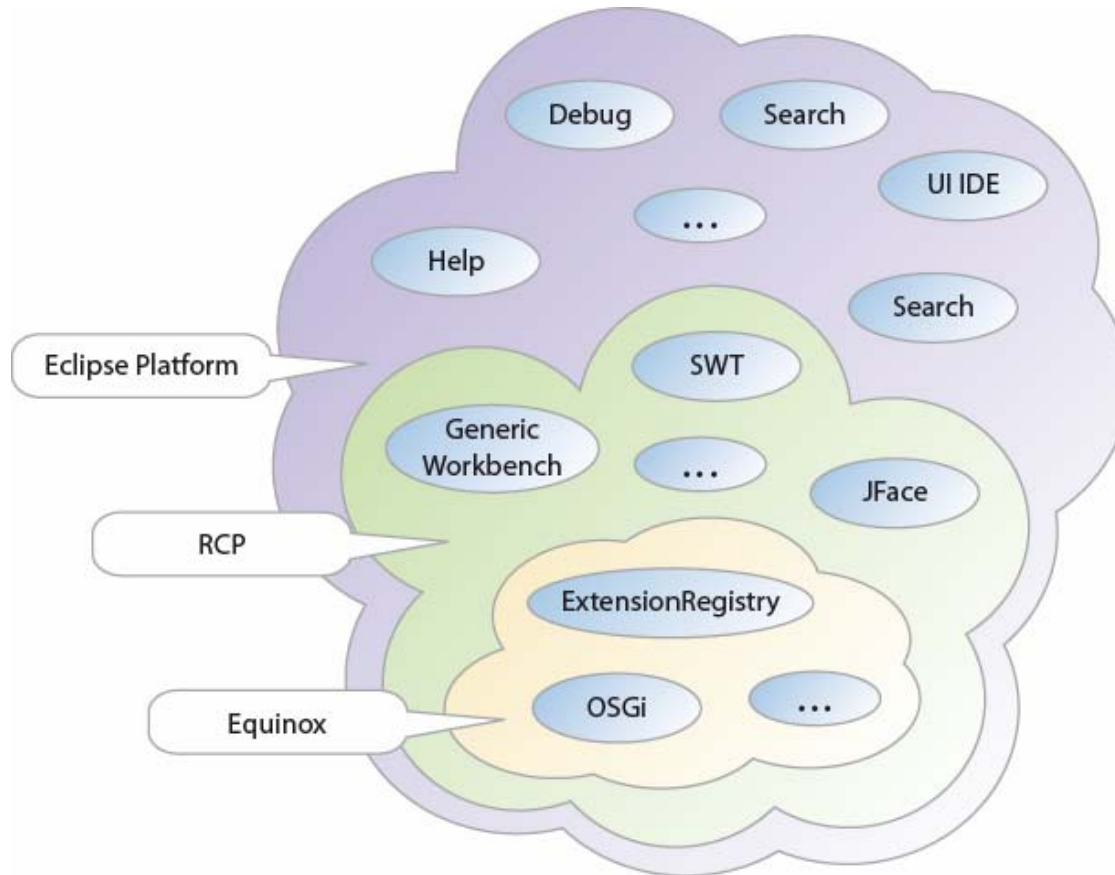
Eclipse everywhere

- Old fashioned:
 - Eclipse is a nice Java-IDE

- Well established:
 - Eclipse is a well-known framework for developing Rich-Client-Applications (see Lotus Notes and many more...)

- But:
 - Most applications don't have just a rich client
 - Some applications don't even have a rich client

Eclipse Equinox





What's next?

- Server-Side Eclipse:
 - Use Eclipse-Equinox as platform for server-side applications
- Why?



Why?

- Modules via OSGi
 - Declared dependencies, versioning, public vs. private APIs, updating, services, ...
- Building flexible architectures via Extension-Points
 - Platform-based development, component model, extensibility
- And much more:
 - Adapters
 - Jobs
 - Preferences
 - Updating



Many interested parties...

- Interested projects...
 - ECF Project
 - Open Healthcare
 - Rich AJAX Platform
 - Eclipse Component Framework
 - Corona Project
 - ...

Web-Apps the Eclipse Way

1. Filter the list of features

2. Select the feature(s) you want

3. Features you need are automatically added

4. Download when ready

The current download size is: 46.99 MB

Download now

<http://yoxos.com/ondemand/>



Different opportunities

- Pure OSGi – Application
 - Open Service Gateway initiative
 - Helps us to manage dependencies
 - At compile time by the IDE
 - At runtime by OSGi itself
 - Install and Uninstall bundles at runtime

- Equinox – Application
 - Part 1: OSGi and the Extension Registry
 - Part 2: The Extension-Registry without OSGi



Different opportunities

- Eclipse Headless
 - OSGi + Extension Registry + Eclipse-Runtime
 - Just a bit more convenience
- Equinox and OSGi inside a web-container
 - Using the Equinox incubator project
- Web server inside OSGi
 - Running a web server as an OSGi bundle
- Spring and Equinox

Pure OSGi

- Descriptor for a bundle

Bundle-Name: Simpleosgi Plug-in

Bundle-SymbolicName: de.kolbware.samples.simpleosgi

Bundle-Version: 1.0.0

Bundle-Activator: de.kolbware.samples.simpleosgi.Activator

Import-Package: org.osgi.framework;**version**="1.3.0"

- Implementation

```
public class Activator implements BundleActivator {  
  
    public void start(BundleContext context) throws Exception {  
        System.out.println("Hello World!!");  
    }  
  
    public void stop(BundleContext context) throws Exception {  
        System.out.println("Goodbye World!!");  
    }  
}
```



Pure OSGi / Equinox Extension Registry

- **Demo**
 - Install
 - Start
 - Stop
 - Uninstall

- **Demo**
 - Extend the OSGi sample to use the extension registry

Extension-Registry without OSGi

- Still work in progress
 - Till now, you'll have to ship the osgi.jar to keep the class-hierarchy consistent
- **Demo**
- This approach is interesting for environments where the special class-loading of OSGi is not possible or leads to many problems
 - E.g. App-Servers without an built-in OSGi container



Eclipse Headless

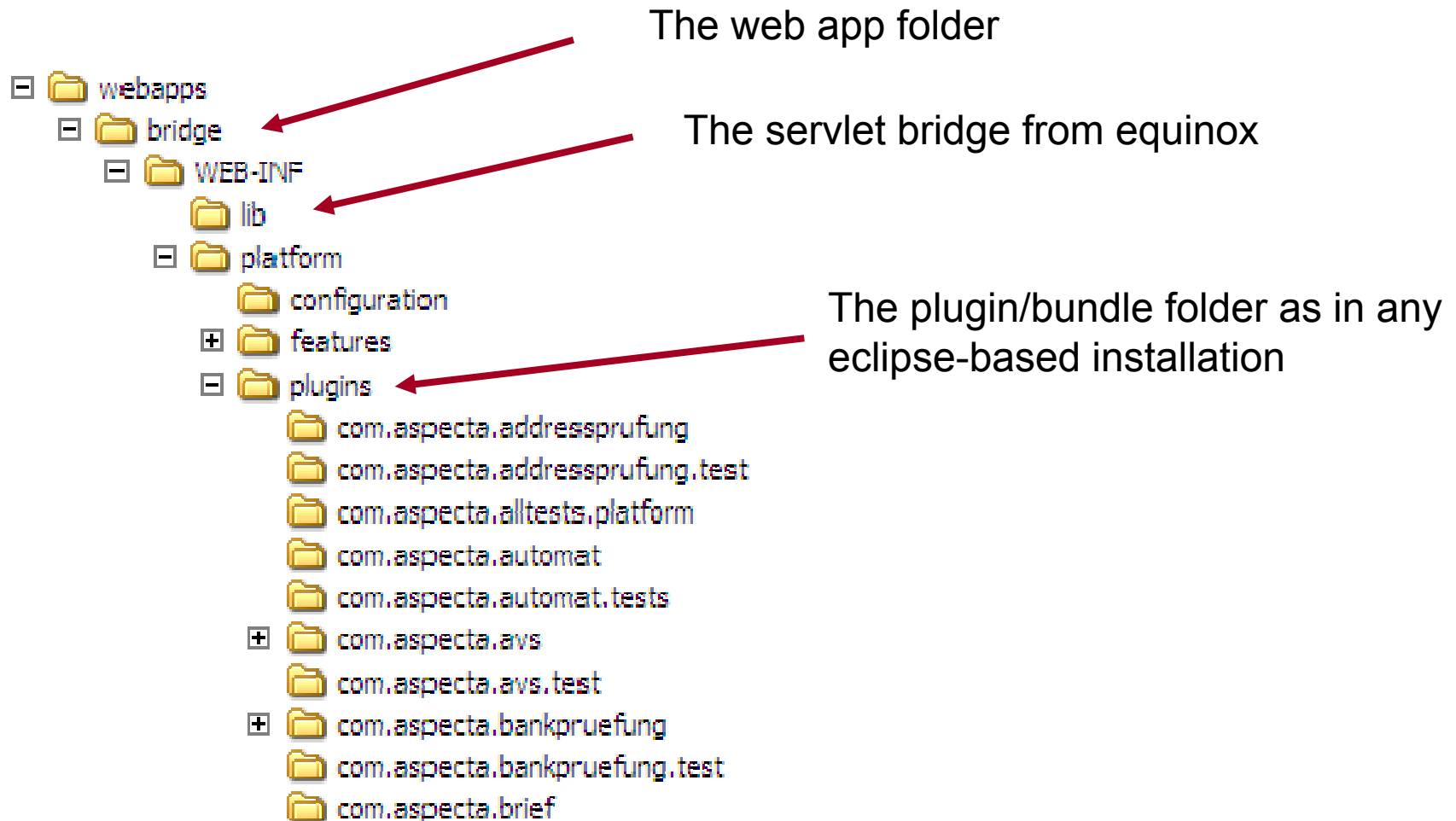
- Same procedure as known from the RCP
 - Implement the Extension-Point
`org.eclipse.core.runtime.applications`
- The Eclipse-Runtime starts our Application
- We can just run one Eclipse-App at the same time.
 - To have several apps, we still can start other bundles manually



OSGi inside a Web-container

- The Equinox incubator project developed a Servlet-Bridge
- The OSGi container is bundled inside a WAR-file
- The Servlet inside the Servlet-Bridge forwards the requests to your servlets
- Servlets and resources can be contributed via an extension point
- **Demo**

The structure of the web-app





Web server in an OSGi container

- The OSGi container starts up normally
- The Web server is wrapped into an OSGi bundle
- A third Plug-In publishes extension-points to register web-apps
- Additionally the servlet bridge can be used
- **Demo**



Spring and OSGi

- Still in development
- The Spring framework is started as an OSGi Bundle
- Other bundles use a subclass of `org.springframework.osgi.context.ContextLoaderBundleActivator`
- The context has to be in the META-INF directory
- The bundle manifests should contain `Eclipse-LazyStart: true`



Spring and OSGi

- **Demo**



Web-Server, OSGi and Spring

- As still in development not everything is working perfectly together
 - Classloading issues
- We will run the Eclipse-Platform inside Jetty using the incubator-code
- We defined a servlet which accesses a spring-service
 - → REST-Based
- **Demo**



Thank you for your attention!

- Questions are welcome!!!

- Further help and assistance:
 - Martin Lippert: martin.lippert@akquinet.de